LITES: een intelligent tutorsysteem voor juridisch onderwijs : een orienterend onderzoek naar de uitgangspunten, opbouw en haalbaarheid van een didactisch expertsysteem

SAMENVATTING

In hoofdstuk 1 hebben wij uiteengezet dat traditionele COO-programmatuur strikte beperkingen kent ten aanzien van *wat* en *hoe* er onderwezen wordt. Wij hebben toen geconcludeerd dat die beperkingen minder knellend zouden zijn, als wij erin zouden slagen een systeem te ontwikkelen dat (enige) kennis heeft van de te onderwijzen materie en de hiervoor te gebruiken didactische methoden. Aangezien expertsystemen gebruik maken van kennis bij het oplossen van problemen, ligt het voor de hand bij de ontwikkeling van intelligente tutorsystemen daarbij aansluiting te zoeken. Wanneer wij expertsystemen als uitgangspunt voor de ontwikkeling van een ITS nemen, betekent dit dat wij van de bij het ontwikkelen van expertsystemen verworven inzichten kunnen profiteren. Deze ontwikkeling maakt gebruik van AI-technieken om verschillende soorten kennis in de computer op te slaan, opdat de computer met behulp van een zogenaamd afleidmechanisme kan redeneren. Maar de AI-technologie heeft, ondanks niet geringe prestaties, niet voor elk probleem een pasklare oplossing. Een van de problemen waarvoor de AI-technologie nog steeds geen oplossing biedt, is de verwerking van natuurlijke taal. De gebruiker kan nog niet op een voldoende hoog niveau met de computer communiceren in zijn eigen taal. Omdat het niet te verwachten is dat hierin binnen afzienbare tijd verbetering komt, is bij dit onderzoek van meet af aan afgezien van de mogelijkheid dat computer en gebruiker in een natuurlijke taal kunnen communiceren. In LITES verloopt de communicatie tussen programma en gebruiker nogal stroef en eenzijdig. Het is een beperking, die wij vooralsnog voor lief nemen.

Het door ons ontwikkelde ITS heeft kennis van het te onderwijzen leerstuk en draagt deze op didactisch verantwoorde wijze over aan de gebruiker. Hierdoor wijkt dit programma sterk af van de opbouw van traditioneel COO, dat weliswaar ook kennis in zich draagt, maar zelf geen enkele vorm van inzicht in die kennis heeft. LITES bezit expliciete kennis en is in staat op basis hiervan conclusies te trekken. Dit rechtvaardigt de pretentie dat het intelligenter is dan traditioneel COO. LITES valt derhalve in de categorie ICOO: intelligent computer ondersteund onderwijs. De grotere intelligentie van LITES in vergelijking met traditioneel COO komt in de volgende zes punten tot uitdrukking.

1. Het systeem geeft minder routinematige en meer gedetailleerde reacties.
2. Het systeem biedt het onderwijsmateriaal in grotere variatie aan.
3. Het systeem heeft een groter diagnostiserend vermogen doordat het fouten kan typeren.
4. Het systeem kan het feitenmateriaal aanpassen op basis van de door de gebruiker gemaakte fouten.
5. Het systeem kan zijn didactische strategie aanpassen aan het niveau van de gebruiker.
6. Het systeem geeft een meer gedetailleerde eindbeoordeling.

In hoofdstuk 2 hebben wij vervolgens een aantal bestaande ITS'en nader bekeken. Daarbij blijken de verschillen tussen de systemen enorm te zijn. Deze verschillen betreffen zowel de wijze waarop de kennis van het leerstuk gerepresenteerd wordt, als de didactische principes die aan de systemen ten grondslag liggen. Op grond daarvan

werd geconcludeerd dat er vooralsnog geen uniforme architectuur voor ITS'en bestaat, zij het dat een aantal componenten in elk ITS zijn te vinden. De belangrijkste daarvan zijn:

- Een kennisbestand, waarin de kennis van het te onderwijzen leerstuk is opgeslagen, aan de hand waarvan de problemen binnen het leerstuk opgelost kunnen worden.
- Een student-model, waarin de prestatie van de gebruiker in de vorm van een 'studenthistorie' wordt opgeslagen.
- Een arsenaal van didactische strategieën, waaruit gekozen wordt, afhankelijk van de behoefte van de gebruiker.
- Een mechanisme dat op grond van de antwoorden en de voorgaande prestaties van de student zowel de vereiste didactische respons kan geven als een aangepaste leerstrategie kan kiezen.

Over de wijze waarop deze componenten in elkaar grijpen en de manier waarop naar de gebruiker gereageerd wordt, bestaat geen consensus. Een reden hiervoor kan zijn, dat de besproken systemen elk op een ander vakgebied betrekking hebben, die elk een eigen didactiek vergen. Als dat het geval is, dan is het vreemd dat ook binnen de door ons bekeken juridische ITS'en een grote verscheidenheid in architectuur bestaat. Dat komt waarschijnlijk door het ontbreken van een vakdidactiek die concreet genoeg is om te implementeren in een computerprogramma. Iedere ontwerper heeft zijn eigen opvattingen en voorkeuren op basis waarvan hij een ITS ontwikkelt.

Onder die omstandigheden leek het ons bij het kiezen van de didactische vorm van de door ons te ontwikkelen computer-tutor een redelijk uitgangspunt om een profiel te schetsen van 'de ideale menselijke tutor' en een poging te ondernemen om de computer-tutor daarnaar te modelleren. Dat is ons maar zeer ten dele gelukt, waarbij wij echter troost hebben geput uit de wetenschap dat ook de ideale menselijke docent niet bestaat. Ten opzichte van een levende docent verkeert de computer in menig opzicht in het nadeel. De computer is niet in staat om in natuurlijke taal te communiceren en kan signalen op het vlak van de non-verbale communicatie niet opvangen laat staan interpreteren. Hoewel op dit gebied de ontwikkeling niet stil blijft staan is het vooralsnog uitgesloten dat dergelijke communicatieve vaardigheden tot de reële mogelijkheden van een computer behoren.

Ook de prestaties van LITES zijn verre van ideaal. LITES maakt gebruik van produktieregels, gebaseerd op propositie-logica, om de kennis van het rechtsgebied en de didactische strategieën te implementeren. De hiervoor ontwikkelde taal PL+ maakt daarbij geen duidelijk onderscheid tussen pure kennisregels en sturingsregels. Met andere woorden, in het regelbestand ligt soms niet alleen kennis opgeslagen, maar ook de manier waarop het afleidmechanisme door deze kennis geleid moet worden (sturing). Daarom kan men beter niet van 'een kennisbestand', maar van 'een regelbestand' spreken.

De kennisrepresentatietaal PL+ en het afleidmechanisme zijn in hoofdstuk 3 gedetailleerd beschreven, zodat men een idee krijgt van de kracht van LITES. LITES is een ITS-shell, hetgeen wil zeggen dat het mogelijk is om verschillende soorten kennis in LITES te modelleren en dus ook verschillende ITS'en in LITES te bouwen. De architectuur van het systeem van de derde verkrijger te goeder trouw kan

daarvoor als voorbeeld genomen worden, maar ook is het mogelijk om deze architectuur op een ander onderwerp toe te passen. De uitvoerige beschrijving van de kennisrepresentatietaal PL+ leek ons dienstig om de reikwijdte van de mogelijkheden van LITES te beoordelen.

Wanneer wij een juridisch leerstuk willen implementeren in LITES voor onderwijskundige doeleinden dan moet dit leerstuk aan de volgende criteria voldoen:
- er moet voldoende *consensus* bestaan omtrent de inhoud en de uitwerking van het leerstuk;
- het leerstuk dient *inzichtelijk en gestructureerd* te zijn, hetgeen wil zeggen dat er duidelijke verbanden dienen te bestaan tussen de afzonderlijke elementen waaruit het leerstuk is opgebouwd;
- het leerstuk dient *separabel* te zijn, dat wil zeggen een duidelijk af te bakenen gebied betreffen, zodat het in voldoende mate beschreven kan worden zonder dat men genoodzaakt is andere, aangrenzende leerstukken te beschrijven.

Hoewel deze criteria er helder uitzien, kan men toch niet onmiddellijk aan ieder juridisch leerstuk zien of het eraan voldoet: dat is een kwestie van proberen en niet te snel opgeven. Voor het overige worden aan het juridische kennisbestand van LITES minder zware eisen gesteld dan aan een juridisch kennisbestand in een expertsysteem. Dit komt doordat er zich bij LITES geen interpretatieproblemen voordoen tussen de aangeboden feiten die de casus beschrijven en de voor het oplossen van de casus gebruikte kennis. Dergelijke interpretatieproblemen doen zich wel voor bij expertsystemen, aangezien daar de feiten door iemand anders worden aangedragen dan de maker van het kennisbestand. In LITES leidt het systeem zelf de casusfeiten af uit de aanwezige kennis.

Men kan tenslotte nog een eis van praktische aard stellen voor de representatie van een leerstuk in PL+ ten behoeve van een ITS:
- het leerstuk moet van een *moeilijkheidsgraad* zijn, waarvoor het de moeite loont om een ITS te maken. Het leerstuk moet derhalve niet zo eenvoudig zijn, dat net zo goed met een normaal, d.w.z. eenvoudiger COO-programma zou kunnen worden volstaan.

Voor de representatie van een leerstuk binnen LITES in de taal PL+ geven wij er uit didactische overwegingen de voorkeur aan uit te gaan van een schematische weergave van het leerstuk. Zo'n schema dient weer te geven hoe casus binnen het leerstuk worden opgelost. Zulk een weergave sluit beter aan bij het doel van het systeem: de student inzicht verschaffen in de opbouw en samenhang van de verschillende componenten van het leerstuk.

Wanneer wij binnen LITES een ITS bouwen, dan bestaat zo'n ITS uit de volgende onderdelen:
- een kennisbestand waarmee een casus binnen het leerstuk opgelost kan worden (expert-kennis);
- een casusgenerator, die binnen het gerepresenteerde leerstuk elke casus kan samenstellen;

- regelbestanden die uitdrukking geven aan verschillende didactische strategieën;
- een bestand dat een afspiegeling vormt van de kennis en de voortgang van de student (studentmodel);
- een regelbestand dat op basis van het studentmodel en de studenthistorie telkens beslist welke didactische strategie gebruikt moet worden in het leerproces van de student.

Hoe een dergelijk ITS verder gebouwd wordt en van welke didactische strategieën gebruik gemaakt wordt, hangt grotendeels af van de inventiviteit en deskundigheid van de maker. In deze dissertatie is in hoofdstuk 4 en 5 slechts één mogelijke manier beschreven, hetgeen niet betekent dat dit de enig mogelijke manier is. Wanneer twee mensen onafhankelijk van elkaar in LITES een ITS over hetzelfde leerstuk zouden bouwen, zouden er waarschijnlijk twee nogal verschillende produkten ontstaan, die slechts met elkaar gemeen zouden hebben dat zij op hetzelfde leerstuk betrekking hadden. Wij beschouwen dat als een voordeel: de meeste COO-drivers vormen een knellend keurslijf voor de cursusontwerper, dat hem weinig bewegingsvrijheid laat en nauwelijks gelegenheid biedt voor eigen ideeën. De bewegingsvrijheid die LITES de ontwerper biedt wordt bepaald door de uitdrukkingskracht van de taal PL+. Die is aanmerkelijk groter dan die van de meeste COO-drivers, maar niet geheel zonder beperkingen. Een extra beperking is dat het regelbestand in PL+ niet groter dan 64 Kbyte kan zijn. Maar de bewegingsvrijheid die LITES biedt heeft ook een nadeel: de cursusontwerper moet zelf nadenken over de opbouw van zijn ITS. Het maken van een ITS is een ingewikkeld proces, dat meer tijd vergt dan het maken van een gewone COO-cursus. Voor men aan de bouw van een ITS begint moet men zich het volgende afvragen:

- Hoe omvangrijk en complex is het kennisdomein?
- Hoe spoedig en vaak worden veranderingen met grote doorwerking in het leerstuk verwacht?
- Is het leerstuk in beweging? Welke consequenties heeft dat voor de opbouw van de regelbestanden opdat veranderingen snel en eenvoudig kunnen worden aangebracht?
- Loont het de moeite en de tijd om het leerstuk in een ITS onder te brengen of levert een ITS weinig meerwaarde in vergelijking met een traditioneel COO-programma over hetzelfde leerstuk?
- Hoe moet de architectuur van het ITS eruitzien: welke didactische principes worden in regelbestanden neergelegd, hoe worden de regelbestanden gekoppeld en welke gegevens moeten onderling uitgewisseld worden?
- Is het een haalbare constructie gezien de complexiteit van het leerstuk, de gewenste didactische methoden, de beperktheid van de kennisrepresentatietaal PL+ en de beperkte omvang van de regelbestanden?
- Kan de ontwerper beschikken over voldoende ondersteuning van zowel vakinhoudelijke als informatie-technische aard?

Uiteindelijk concluderen wij (hoofdstuk 6) dat wij in onze doelstelling, het bouwen van een juridisch ITS, geslaagd zijn. In LITES zijn de eerder beschreven 6 programmavormen van intelligentie terug te vinden en hierdoor is LITES intelligenter dan normale COO-programma's.

SUMMARY

In the first chapter it is explained that existing CAI-programs usually are rather restricted in *what* and *how* they teach. Constructing a system which has knowledge of the subject matter it teaches and of the teaching methods it employs would constitute a considerable advantage. Since expert systems are already available which use knowledge in finding solutions to problems, it seems obvious that in trying to build intelligent tutoring systems (ITS) we draw on methods used in developing expert systems. Expert systems use AI-techniques to store various kinds of knowledge in the computer, allowing the computer to reason with this knowledge. AI-technology, however, does not have a solution for every problem. One of the problems that AI-technology cannot yet handle is the parsing and understanding of natural language. The computer-user is not able to communicate with the computer in his own language with any degree of sophistication. Since we do not expect this to change in the near future, at an early stage of our work we abandoned all plans to built a system in which the users can communicate with the computer in natural language. In LITES the communication between the program and its user is rather limited and one-sided. As yet we see no other way.

The ITS we developed has knowledge of the legal doctrine it teaches and passes it to the user in a responsible didactic manner. For this reason, its structure is considerably different from the structure of traditional CAI-programs. Of course, such CAI-programs do have some built-in didactic knowledge, but they have no 'insight' into that knowledge. LITES, on the other hand, has explicit knowledge from which conclusions may be drawn. This justifies our claim that our program and programs like it are more intelligent than traditional CAI-programs. LITES therefore belongs to the class of ICAI-programs: intelligent computer assisted instruction programs. The intelligence we claim for LITES finds its expression in the following six points:

1. The system responds to its user in a less routinized and more detailed manner.
2. The system allows more variety in the teaching materials.
3. The system has more diagnostic power, because it can classify errors.
4. The system is able to adjust the content of the teaching materials offered to the errors made by its users.
5. The system is able to adapt its didactic strategies to the students' level of understanding.
6. The system is able to inform its users in a detailed manner about their performance.

Chapter 2 contains a review of existing intelligent tutoring systems. The examined systems turn out to be vastly different. These differences concern the way the knowledge of the subject matter is represented, as well as the didactic principles underlying the systems. At the moment there appears to be no uniform architecture for intelligent tutoring systems, although they have some basic components in common. The most important common features are:

- A knowledge base, containing knowledge about some doctrine, enabling the system to solve cases coming under that doctrine.
- A student-model, monitoring student performance in the form of a student history.

- A set of didactic strategies from which the system can choose on the basis of the students' needs.
- A mechanism that can give the right didactic response and can choose an adapted teaching-strategy based upon the students prior results.

However, there appears to be no consensus as to the way in which these components should be linked together and respond to the student. The major reason for this lack of consensus may be that each system deals with a different domain of expertise, each with its own methodology. If this is indeed the major reason, one may wonder why it is that also among intelligent tutoring systems within the legal domain such a wide variety of architectures is found. This may be because the law has a comparatively weak methodology, i.e. a methodology precise enough to implement into a computer program. It shows that every designer has his own ideas and preferences in building an intelligent tutoring system.

Given these circumstances, we thought it wise to develop a profile of 'the ideal human tutor' and take this as a starting-point for modelling an intelligent tutoring system. In this we only succeeded partially, but we derive comfort from the knowledge that 'the ideal human tutor' does not exist either. In comparison with a live, human tutor the computer is in many respects at a disadvantage. Computers are not (yet) able to communicate in natural language with their users and also cannot pick up signals of non-verbal communication; to say nothing of understanding them. Those are capabilities of major didactic importance in live tutors. Although some progress in these areas is still made, it is, as yet, out of the question that these aspects of communication can be modelled on computers in the foreseeable future.

Therefore also the performance of LITES falls short of the performance of 'the ideal human tutor'. LITES uses production-rules, based on the propositional calculus, to implement domain-knowledge and its didactic strategies. PL+, the language we developed especially for this purpose, does not distinguish between knowledge-rules and guidance-rules. In other words, the knowledge bases do not only contain knowledge of the subject matter, but also information on the way in which the inference engine has to derive conclusions from the knowledge of the subject matter (guidelines for the inference engine). For this reasons it would be better not to speak of knowledge bases, but rather of rule bases.

The knowledge representation language PL+ and the inference engine are describe in detail in chapter 3, enabling the reader to form his own impression of the power of LITES. LITES is an ITS-shell, which expresses the possibility for modelling various kinds of knowledge within LITES and therefore developing different kinds of intelligent tutoring systems. In this book we describe the architecture of an ITS based upon the 'third party in good faith'-doctrine. It may be taken as a prototype for constructing new tutoring systems within the legal field.

If one wishes to implement some legal doctrine as learning material in LITES, the doctrine must meet the following criteria:

- there must be sufficient consensus about the purport of that doctrine and its width of application;

- the doctrine must be clear and well-structured, i.e. there must be a clear relationship between the various parts of the doctrine;
- the doctrine must be clearly separable, which makes it possible to describe it without referring to an adjacent doctrine.

Although these criteria may look plain enough, it is not apparent without further ado whether a doctrine meets these requirements. It is just a matter of trial and error with a dose of perseverance. On the other hand, the legal knowledge bases in LITES meet fewer requirements than legal knowledge bases in expert systems. That is because in LITES there are no interpretation problems with regard to the facts describing a case and those used to solve the case. Problems of interpretation frequently arise in expert systems, because the facts describing a case are supplied by the user, while the facts used to solve the case come from the system. In LITES the system itself derives its case-facts from its knowledge bases.

Finally we add a criterium of practical value for representing a doctrine in LITES:
- the doctrine has to be of a degree of complexity to make it worthwhile to build an ITS. That means the doctrine cannot be of such a low level of difficulty that a normal CAI-program would suffice.

For representing a particular doctrine in LITES it is practical to start from a schematic representation or flow-chart of the doctrine. Such a diagram shows how cases are solved under the doctrine. This idea links up with the aim of the system, i.e. providing insight into the structure of the doctrine and the connections between its components.

An ITS within LITES consists of the following components:
- a knowledge base, which can solve cases within the doctrine (expert-knowledge);
- a case-generator, which generates cases which can be solved under the doctrine;
- rule bases expressing the didactic strategies used within the system;
- a database representing the students' knowledge and progress (student-model);
- a rule base for determining, on bases of the student-model and the student-history, which didactic strategy to use.

How to make such an ITS and which didactic strategies to use, depends largely on the experience and inventiveness of the designer. Chapters 4 and 5 present only one possible way of building an ITS in LITES. This does not mean that this is the only way possible. Two designers working independently to construct an ITS for the same doctrine, would probably end up with two entirely different tutoring systems. We consider this as an advantage of LITES, because many CAI-drivers are like a straightjacket for the course-designer, giving little room for personal ideas. The freedom LITES leaves to the designer depends on the power of expression of the language PL+. Its capabilities are considerably greater than those of most CAI-languages, but, of course, they are not without limitations. For instance, a rule base in LITES cannot exceed the 64 Kbyte limit. The freedom LITES affords the course-designer can also be considered a disadvantage, because it offers the designer no prefab modules. Each designer is required to think thoroughly about the construction

of the tutoring system. Building an ITS is a complicated task, demanding more time than constructing a normal CAI-course. Before starting to build an ITS, one must ask oneself the following questions:

- How extensive and complex is the knowledge domain?
- How consistent is the knowledge domain?
- How likely is it that there will be significant changes in the domain in the future and to what extend will changes influence the knowledge-bases?
- Is it worthwhile building an ITS for this subject matter, or would a normal CAI-course do for this subject matter?
- What will the architecture of the ITS be, which didactic principles will need to be constructed into rule bases, how will the rule bases be joined together and which data will have to be exchanged between them?
- Is this a feasible construction, considering the complexity of the doctrine, the desired didactic methods and the restrictions of the knowledge representation language PL+?
- Will the designer have sufficient and substantial support, technically and with respect to expertise?

In chapter 6 it is concluded that we succeeded in constructing an intelligent tutoring system. In LITES it is possible to construct systems that posses the six forms of intelligence mentioned earlier. Thus defined, LITES is more intelligent than normal CAI-programs.